Python for data analysis

July 15, 2024

1 TASK #1: DEFINE SINGLE AND MULTI-DIMENSIONAL NUMPY ARRAYS

[1]:
```
# NumPy is a Linear Algebra Library used for multidimensional arrays
# NumPy brings the best of two worlds: (1) C/Fortran computational efficiency,
 ↪(2) Python language easy syntax
import numpy as np
# Let's define a one-dimensional array
list_1 = [50, 60, 80, 100, 200, 300, 500, 600]
list_1
```

[1]:
```
[50, 60, 80, 100, 200, 300, 500, 600]
```

2 Let's create a numpy array from the list "my_list"

[2]:
```
my_numpy_array = np.array(list_1)
my_numpy_array
```

[2]:
```
array([ 50,  60,  80, 100, 200, 300, 500, 600])
```

[3]:
```
# Multi-dimensional (Matrix definition)
```

MINI CHALLENGE #1: - Write a code that creates the following 2x4 numpy array

```
[[3 7 9 3]
[4 3 2 2]]
```

[4]:
```
m = np.array ([[3,7,9,3],[4,3,2,2]])
m
```

[4]:
```
array([[3, 7, 9, 3],
       [4, 3, 2, 2]])
```

3 TASK #2: LEVERAGE NUMPY BUILT-IN METHODS AND FUNCTIONS

[5]:
```
# "rand()" uniform distribution between 0 and 1
x = np.random.rand(20)
x
```

[5]:
```
array([0.94266928, 0.02552089, 0.05226369, 0.62574019, 0.91471137,
       0.97912726, 0.61211093, 0.58036854, 0.32697877, 0.94282886,
       0.42167133, 0.53247038, 0.11237543, 0.19895323, 0.15727569,
       0.4309179 , 0.55585634, 0.13845179, 0.83527666, 0.23828949])
```

[6]:
```
# you can create a matrix of random number as well
x = np.random.rand(3,3)
x
```

[6]:
```
array([[0.95443371, 0.29158316, 0.44011282],
       [0.78617126, 0.25759616, 0.18953674],
       [0.7290729 , 0.0079873 , 0.31086896]])
```

[7]:
```
# "randint" is used to generate random integers between upper and lower bounds
x = np.random.randint(1,50)
x
```

[7]:
```
38
```

[8]:
```
# "randint" can be used to generate a certain number of random itegers as␣
↪follows
x = np.random.randint(1,100,15)
x
```

[8]:
```
array([62, 72, 49,  2,  9, 84, 93, 11, 45, 59,  8, 41, 47, 84, 10])
```

[9]:
```
# np.arange creates an evenly spaced values within a given interval
x = np.arange(1,50)
x
```

[9]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])
```

[10]:
```
# create a diagonal of ones and zeros everywhere else
x = np.eye(7)
x
```

[10]:
```
array([[1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0.],
```

```
       [0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 1.]])
```

[11]:
```
# Matrix of ones
x = np.ones((7,7))
x
```

[11]:
```
array([[1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1.]])
```

[12]:
```
# Array of zeros
x = np.zeros(8)
x
```

[12]:
```
array([0., 0., 0., 0., 0., 0., 0., 0.])
```

MINI CHALLENGE #2: - Write a code that takes in a positive integer "x" from the user and creates a 1x10 array with random numbers ranging from 0 to "x"

[ ]:
```
x=int(input('Please enter a positive value'))
0
```

[ ]:
```
x = np.random.randint(1,x,10)
0
```

4 TASK #3: PERFORM MATHEMATICAL OPERATIONS IN NUMPY

[15]:
```
# np.arange() returns an evenly spaced values within a given interval
x = np.arange(1,10)
x
```

[15]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[16]:
```
y =np.arange(1,10)
y
```

[16]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[17]:
```
# Add 2 numpy arrays together
sum = x+y
sum
```

[17]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

[18]:
```
squared=x**2
squared
```

[18]:
```
array([ 1,  4,  9, 16, 25, 36, 49, 64, 81])
```

[19]:
```
sqrt= np.sqrt(squared)
sqrt
```

[19]:
```
array([1., 2., 3., 4., 5., 6., 7., 8., 9.])
```

[20]:
```
z = np.exp(y)
z
```

[20]:
```
array([2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500e+01,
       1.48413159e+02, 4.03428793e+02, 1.09663316e+03, 2.98095799e+03,
       8.10308393e+03])
```

MINI CHALLENGE #3: - Given the X and Y values below, obtain the distance between them

```
X = [5, 7, 20]
Y = [9, 15, 4]
```

[21]:
```
x = np.array([5, 7, 20])
y = np.array([9, 15, 4])

z = np.sqrt(x**2 + y**2 )
z
```

[21]:
```
array([10.29563014, 16.55294536, 20.39607805])
```

5 TASK #4: PERFORM ARRAYS SLICING AND INDEXING

[22]:
```
import numpy as np
```

[23]:
```
my= np.array([3, 5, 6, 2, 8, 10, 20, 50])
my
```

[23]:
```
array([ 3,  5,  6,  2,  8, 10, 20, 50])
```

[24]:
```
# Access specific index from the numpy array
my[5]
```

[24]:
```
10
```

[25]:
```
# Starting from the first index 0 up until and NOT including the last element
my[0:7]
```

[25]:
```
array([ 3,  5,  6,  2,  8, 10, 20])
```

[26]:
```
# Broadcasting, altering several values in a numpy array at once
my[0:2]=7
my
```

[26]:
```
array([ 7,  7,  6,  2,  8, 10, 20, 50])
```

[27]:
```
# Let's define a two dimensional numpy array
matrix = np.random.randint(1,10,(4,4))
matrix
```

[27]:
```
array([[8, 4, 9, 5],
       [8, 1, 2, 9],
       [8, 8, 8, 3],
       [2, 7, 2, 7]])
```

[28]:
```
# Get a row from a mtrix
matrix[0]
```

[28]:
```
array([8, 4, 9, 5])
```

[29]:
```
# Get one element
matrix[0][2]
```

[29]:
```
9
```

MINI CHALLENGE #4: - In the following matrix, replace the last row with 0

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[30]:
```
import numpy as np
```

[31]:
```
x= np.array ([[2, 30, 20, -2, -4],
              [3, 4,  40, -3, -2],
              [-3, 4, -6, 90, 10],
              [25, 45, 34, 22, 12],
              [13, 24, 22, 32, 37]])
x[4]=0
```

```
x
```

[31]:
```
array([[ 2, 30, 20, -2, -4],
       [ 3,  4, 40, -3, -2],
       [-3,  4, -6, 90, 10],
       [25, 45, 34, 22, 12],
       [ 0,  0,  0,  0,  0]])
```

6 TASK #5: PERFORM ELEMENTS SELECTION (CONDITIONAL)

[32]:
```
matrix = np.random.randint(1,10, (5,5))
matrix
```

[32]:
```
array([[1, 4, 5, 5, 3],
       [4, 4, 8, 6, 1],
       [1, 3, 6, 6, 3],
       [9, 9, 1, 6, 9],
       [7, 7, 8, 9, 6]])
```

[33]:
```
new_matrix = matrix[matrix > 7 ]
new_matrix
```

[33]:
```
array([8, 9, 9, 9, 8, 9])
```

[34]:
```
# Obtain odd elements only
new_matrix = matrix[matrix % 2 ==1]
new_matrix
```

[34]:
```
array([1, 5, 5, 3, 1, 1, 3, 3, 9, 9, 1, 9, 7, 7, 9])
```

MINI CHALLENGE #5: - In the following matrix, replace negative elements by 0 and replace odd elements with -2

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[35]:
```
X = np.array([[2, 30, 20, -2, -4],
              [3, 4,  40, -3, -2],
              [-3, 4, -6 ,90, 10],
              [25, 45, 34, 22, 12],
              [13, 24, 22, 32, 37]])
x[x<0]=0
x[x%2==1]=-2
```

```
x
```

[35]:
```
array([[ 2, 30, 20,  0,  0],
       [-2,  4, 40,  0,  0],
       [ 0,  4,  0, 90, 10],
       [-2, -2, 34, 22, 12],
       [ 0,  0,  0,  0,  0]])
```

7 TASK #6: UNDERSTAND PANDAS FUNDAMENTALS

[36]:
```
# Pandas is a data manipulation and analysis tool that is built on Numpy.
# Pandas uses a data structure known as DataFrame (think of it as Microsoft excel in Python).
# DataFrames empower programmers to store and manipulate data in a tabular fashion (rows and columns).
# Series Vs. DataFrame? Series is considered a single column of a DataFrame.
```

[37]:
```
import pandas as pd
import numpy as np
```

[38]:
```
# Let's define a two-dimensional Pandas DataFrame
# Note that you can create a pandas dataframe from a python dictionary
bank_dataf = pd.DataFrame({'Bank client ID':[111,222,333,444],
                           'Bank client Name':['Doran', 'Steve','Mitch', 'Ryan'],
                           'Net Worth[$]':[3500,2900,1000,2000],
                           'Years with Bank':[3, 4, 9, 5]})
print(bank_dataf)
```

```
   Bank client ID Bank client Name  Net Worth[$]  Years with Bank
0             111            Doran          3500                3
1             222            Steve          2900                4
2             333            Mitch          1000                9
3             444             Ryan          2000                5
```

[39]:
```
# Let's obtain the data type

type(bank_dataf)
```

[39]:
```
pandas.core.frame.DataFrame
```

[40]:
```
# you can only view the first couple of rows using .head()
bank_dataf.head()
```

[40]:
```
   Bank client ID Bank client Name  Net Worth[$]  Years with Bank
0             111            Doran          3500                3
1             222            Steve          2900                4
```

```
2              333            Mitch          1000                9
3              444             Ryan          2000                5
```

[41]:
```
# you can only view the last couple of rows using .tail()
bank_dataf.tail()
```

[41]:
```
   Bank client ID Bank client Name  Net Worth[$]  Years with Bank
0             111            Doran          3500                3
1             222            Steve          2900                4
2             333            Mitch          1000                9
3             444             Ryan          2000                5
```

MINI CHALLENGE #6: - A porfolio contains a collection of securities such as stocks, bonds and ETFs. Define a dataframe named 'portfolio_df' that holds 3 different stock ticker symbols, number of shares, and price per share (feel free to choose any stocks) - Calculate the total value of the porfolio including all stocks

[42]:
```
portfolio_df = pd.DataFrame({'stock ticker':['SPY', 'QQQ','SOFI'],
                             'number of shares':[20,25,30],
                             'PPS':[556,495,7]})
portfolio_df
stock_dollar_value = portfolio_df['number of shares'] * portfolio_df['PPS']
stock_dollar_value
```

[42]:
```
0    11120
1    12375
2      210
dtype: int64
```

8 TASK #7: PANDAS WITH CSV AND HTML DATA

[43]:
```
# Pandas is used to read a csv file and store data in a DataFrame
import pandas as pd
```

[44]:
```
data_entry_salary = pd.read_html('https://www.livingin-canada.com/
 ↪salaries-for-data-entry-clerks-canada.html')
```

[45]:
```
!pip install lxml
!python -m pip install --upgrade pip
```

```
Requirement already satisfied: lxml in /opt/conda/lib/python3.10/site-packages
(5.2.2)
Requirement already satisfied: pip in /opt/conda/lib/python3.10/site-packages
(24.1.2)
```

[46]:
```
data_entry_salary[0]
```

[46]:

	Location	Low Wage $ per hr	Average Wage $ per hr
0	Calgary - Alberta*	15.00	22.48
1	Edmonton - Alberta*	17.40	20.77
2	Vancouver / Lower Mainland Southwest - British…	14.00	20.00
3	NaN	NaN	NaN
4	Winnipeg - Manitoba	11.65	18.38
5	Fredericton / Oromocto - New Brunswick	11.50	21.00
6	Halifax - Nova Scotia	12.05	
7	NaN	NaN	
8	Toronto - Ontario	14.00	
9	Ottawa - Ontario	14.00	
10	Hamilton / Niagara Peninsula - Ontario	14.00	
11	NaN	NaN	
12	Windsor / Sarnia Region - Ontario(1)	14.00	
13	Prince Edward Island	14.53	
14	Montreal - Quebec	15.00	
15	NaN	NaN	
16	Saskatoon / Biggar - Saskatchewan	15.00	
17	(adsbygoogle = window.adsbygoogle || []).push(…	(adsbygoogle = window.adsbygoogle || []).push(…	

```
6                                                 19.23
7                                                   NaN
8                                                 20.00
9                                                 20.19
10                                                20.00
11                                                  NaN
12                                                20.00
13                                                20.00
14                                                16.00
15                                                  NaN
16                                                20.67
17  (adsbygoogle = window.adsbygoogle || []).push(…

                                    High Wage $ per hr  \
0                                                 34.04
1                                                 27.67
2                                                 32.69
3                                                   NaN
4                                                 30.01
5                                                 26.00
6                                                 23.50
7                                                   NaN
8                                                 31.67
9                                                 32.31
10                                                25.60
11                                                  NaN
12                                                31.25
13                                                23.65
14                                                25.84
15                                                  NaN
16                                                25.13
17  (adsbygoogle = window.adsbygoogle || []).push(…

                                                   Year
0                                                  2019
1                                                  2019
2                                                  2018
3                                                   NaN
4                                                  2018
5                                                  2018
6                                                  2018
7                                                   NaN
8                                                  2018
9                                                  2018
10                                                 2018
11                                                  NaN
12                                                 2018
```

```
13                                                 2018
14                                                 2018
15                                                  NaN
16                                                 2018
17  (adsbygoogle = window.adsbygoogle || []).push(…
```

[47]:
```
# Read tabular data using read_html
```

[48]:
```
import pandas as pd
retirement_data = pd.read_html('https://www.ssa.gov/oact/progdata/nra.html')
```

[49]:
```
retirement_data[0]
```

[49]:
```
                                    Year of birth  \
0                                  1937 and prior
1                                            1938
2                                            1939
3                                            1940
4                                            1941
5                                            1942
6                                         1943-54
7                                            1955
8                                            1956
9                                            1957
10                                           1958
11                                           1959
12                                  1960 and later
13  Notes: 1. Persons born on January 1 of any yea…

                                             Age
0                                             65
1                                  65 and 2 months
2                                  65 and 4 months
3                                  65 and 6 months
4                                  65 and 8 months
5                                 65 and 10 months
6                                             66
7                                  66 and 2 months
8                                  66 and 4 months
9                                  66 and 6 months
10                                 66 and 8 months
11                                66 and 10 months
12                                            67
13  Notes: 1. Persons born on January 1 of any yea…
```

MINI CHALLENGE #7: - Write a code that uses Pandas to read tabular US retirement data - You can use data from here: https://www.ssa.gov/oact/progdata/nra.html

[50]:
```
retirement_data[0]
```

[50]:

	Year of birth	\
0	1937 and prior	
1	1938	
2	1939	
3	1940	
4	1941	
5	1942	
6	1943-54	
7	1955	
8	1956	
9	1957	
10	1958	
11	1959	
12	1960 and later	
13	Notes: 1. Persons born on January 1 of any yea…	

	Age
0	65
1	65 and 2 months
2	65 and 4 months
3	65 and 6 months
4	65 and 8 months
5	65 and 10 months
6	66
7	66 and 2 months
8	66 and 4 months
9	66 and 6 months
10	66 and 8 months
11	66 and 10 months
12	67
13	Notes: 1. Persons born on January 1 of any yea…

9 TASK #8: PANDAS OPERATIONS

[51]:
```
# Let's define a dataframe as follows:
bank_dataf = pd.DataFrame({'Bank client ID':[111,222,333,444],
                           'Bank client Name':['Doran', 'Steve','Mitch', 'Ryan'],
                           'Net Worth[$]':[3500,29000,10000,2000],
                           'Years with Bank':[3, 4, 9, 5]})
bank_dataf
```

[51]:

	Bank client ID	Bank client Name	Net Worth[$]	Years with Bank
0	111	Doran	3500	3
1	222	Steve	29000	4

```
2              333             Mitch          10000                 9
3              444              Ryan           2000                 5
```

[52]:
```
# Pick certain rows that satisfy a certain criteria
df_loyal = bank_dataf[bank_dataf['Years with Bank']>=5]
df_loyal
```

[52]:
```
   Bank client ID Bank client Name  Net Worth[$]  Years with Bank
2              333             Mitch          10000                 9
3              444              Ryan           2000                 5
```

[53]:
```
# Delete a column from a DataFrame
del bank_dataf['Bank client ID']
bank_dataf
```

[53]:
```
  Bank client Name  Net Worth[$]  Years with Bank
0            Doran          3500                 3
1            Steve         29000                 4
2            Mitch         10000                 9
3             Ryan          2000                 5
```

MINI CHALLENGE #8: - Using "bank_client_df" DataFrame, leverage pandas operations to only select high networth individuals with minimum $5000 - What is the combined networth for all customers with 5000+ networth?

[54]:
```
high_networth = bank_dataf[bank_dataf['Net Worth[$]']>=5000]
high_networth
```

[54]:
```
  Bank client Name  Net Worth[$]  Years with Bank
1            Steve         29000                 4
2            Mitch         10000                 9
```

[55]:
```
high_networth['Net Worth[$]'].sum()
```

[55]:
```
39000
```

10 TASK #9: PANDAS WITH FUNCTIONS

[56]:
```
# Let's define a dataframe as follows:
import pandas as pd
bank_client_df = pd.DataFrame({'Bank client ID':[111, 222, 333, 444],
                               'Bank Client Name':['Chanel', 'Steve', 'Mitch',␣
 ↪'Ryan'],
                               'Net worth [$]':[3500, 29000, 10000, 2000],
                               'Years with bank':[3, 4, 9, 5]})
bank_client_df
```

[56]:

	Bank client ID	Bank Client Name	Net worth [$]	Years with bank
0	111	Chanel	3500	3
1	222	Steve	29000	4
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[57]:
```
# Define a function that increases all clients networth (stocks) by a fixed␣
 ↪value of 20% (for simplicity sake)
def networth_update(balance):
    return balance *1.2
```

[58]:
```
# You can apply a function to the DataFrame
bank_client_df['Net worth [$]'].apply(networth_update)
```

[58]:
```
0     4200.0
1    34800.0
2    12000.0
3     2400.0
Name: Net worth [$], dtype: float64
```

[ ]:
```

```

MINI CHALLENGE #9: - Define a function that triples the stock prices and adds $200 - Apply the function to the DataFrame - Calculate the updated total networth of all clients combined

[59]:
```
def networth_update(balance):
    return balance *3 +200
```

[60]:
```
bank_client_df['Net worth [$]'].apply(networth_update)
```

[60]:
```
0    10700
1    87200
2    30200
3     6200
Name: Net worth [$], dtype: int64
```

11 TASK #10: PERFORM SORTING AND ORDERING IN PANDAS

[61]:
```
# Let's define a dataframe as follows:
bank_client_df = pd.DataFrame({'Bank client ID':[111, 222, 333, 444],
                               'Bank Client Name':['Chanel', 'Steve', 'Mitch',␣
 ↪'Ryan'],
                               'Net worth [$]':[3500, 29000, 10000, 2000],
                               'Years with bank':[3, 4, 9, 5]})
bank_client_df
```

[61]:

	Bank client ID	Bank Client Name	Net worth [$]	Years with bank
0	111	Chanel	3500	3
1	222	Steve	29000	4
2	333	Mitch	10000	9
3	444	Ryan	2000	5

[62]:
```
# You can sort the values in the dataframe according to number of years with␣
↪bank
bank_client_df.sort_values(by ='Years with bank')
```

[62]:

	Bank client ID	Bank Client Name	Net worth [$]	Years with bank
0	111	Chanel	3500	3
1	222	Steve	29000	4
3	444	Ryan	2000	5
2	333	Mitch	10000	9

[63]:
```
# Note that nothing changed in memory! you have to make sure that inplace is␣
↪set to True
```

[64]:
```
# Set inplace = True to ensure that change has taken place in memory
bank_client_df.sort_values(by='Years with bank',inplace =True)
```

[65]:
```
# Note that now the change (ordering) took place
```

12 TASK #11: PERFORM CONCATENATING AND MERGING WITH PANDAS

[66]:
```
# Check this out: https://pandas.pydata.org/pandas-docs/stable/user_guide/
↪merging.html
```

[67]:
```
df1=pd.DataFrame({'A':['A0','A1','A2','A3'],
                  'B':['B0','B1','B2','B3'],
                  'C':['C0','C1','C2','C3'],
                  'D':['D0','D1','D2','D3']})
```

[68]:
```
df1
```

[68]:

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

[69]:
```
df2=pd.DataFrame({'A':['A4','A5','A6','A7'],
                  'B':['B4','B5','B6','B7'],
                  'C':['C4','C5','C6','C7'],
```

```
                    'D':['D4','D5','D6','D7']},
                   index = [4,5,6,7])
```

[70]:
```
df2
```

[70]:
```
     A   B   C   D
4   A4  B4  C4  D4
5   A5  B5  C5  D5
6   A6  B6  C6  D6
7   A7  B7  C7  D7
```

pd.concat([df1,df2])

[ ]:
```

```

[ ]:
```

```

13 TASK #12: PROJECT AND CONCLUDING REMARKS

- Define a dataframe named 'Bank_df_1' that contains the first and last names for 5 bank clients with IDs = 1, 2, 3, 4, 5
- Assume that the bank got 5 new clients, define another dataframe named 'Bank_df_2' that contains a new clients with IDs = 6, 7, 8, 9, 10
- Let's assume we obtained additional information (Annual Salary) about all our bank customers (10 customers)
- Concatenate both 'bank_df_1' and 'bank_df_2' dataframes
- Merge client names and their newly added salary information using the 'Bank Client ID'
- Let's assume that you became a new client to the bank
- Define a new DataFrame that contains your information such as client ID (choose 11), first name, last name, and annual salary.
- Add this new dataframe to the original dataframe 'bank_df_all'.

[71]:
```
import pandas as pd
```

[72]:
```
bank_df1 = pd.DataFrame({'Bank client ID':['1','2','3','4','5'],
                         'First Name':['Mike','Sarah','Tim','Doran','Pam'],
                         'Last Name':
 ↪['Cook','James','Fista','Hamilton','Greer']})


bank_1 = pd.DataFrame(bank_df1,columns = ['Bank client ID', 'First Name','Last␣
 ↪Name'])

bank_1
```

[72]:
	Bank client ID	First Name	Last Name
0	1	Mike	Cook
1	2	Sarah	James
2	3	Tim	Fista
3	4	Doran	Hamilton
4	5	Pam	Greer

[73]:
```
bank_df2 = pd.DataFrame({'Bank client ID':['6','7','8','9','10'],
                         'First Name':['Tom','Bill','Jim','Dora','Sam'],
                         'Last Name':['Crook','Nick','Brooks','Jade','Smith']})


bank_2 = pd.DataFrame(bank_df2,columns = ['Bank client ID', 'First Name','Last␣
 ↪Name'])

bank_2
```

[73]:
	Bank client ID	First Name	Last Name
0	6	Tom	Crook
1	7	Bill	Nick
2	8	Jim	Brooks
3	9	Dora	Jade
4	10	Sam	Smith

[74]:
```
raw_data = {'Bank client ID':['1','2','3','4','5','6','7','8','9','10'],
            'Annual Salary':[35000, 85000, 55000, 85000, 95000, 45000, 66000,␣
 ↪50000, 42000, 37000]}

bank_df_salary = pd.DataFrame(raw_data, columns = ['Bank client ID', 'Annual␣
 ↪Salary'])
bank_df_salary
```

[74]:
	Bank client ID	Annual Salary
0	1	35000
1	2	85000
2	3	55000
3	4	85000
4	5	95000
5	6	45000
6	7	66000
7	8	50000
8	9	42000
9	10	37000

[75]:
```
bank_df_all = pd.concat([bank_1, bank_2])
bank_df_all
```

[75]:

	Bank client ID	First Name	Last Name
0	1	Mike	Cook
1	2	Sarah	James
2	3	Tim	Fista
3	4	Doran	Hamilton
4	5	Pam	Greer
0	6	Tom	Crook
1	7	Bill	Nick
2	8	Jim	Brooks
3	9	Dora	Jade
4	10	Sam	Smith

[76]:
```
bank_df_all =pd.merge(bank_df_all, bank_df_salary, on ='Bank client ID')
bank_df_all
```

[76]:

	Bank client ID	First Name	Last Name	Annual Salary
0	1	Mike	Cook	35000
1	2	Sarah	James	85000
2	3	Tim	Fista	55000
3	4	Doran	Hamilton	85000
4	5	Pam	Greer	95000
5	6	Tom	Crook	45000
6	7	Bill	Nick	66000
7	8	Jim	Brooks	50000
8	9	Dora	Jade	42000
9	10	Sam	Smith	37000

[77]:
```
new_client = {'Bank client ID':['11'],
              'First Name':['Marland'],
              'Last Name':['Hamilton'],
              'Annual Salary':[72000]}
new_client_df =pd.DataFrame(new_client, columns = ['Bank client ID', 'First␣
 ↪Name','Last Name', 'Annual Salary'])
```

[78]:
```
new_client_df
```

[78]:

	Bank client ID	First Name	Last Name	Annual Salary
0	11	Marland	Hamilton	72000

[79]:
```
new_df = pd.concat([bank_df_all, new_client_df], axis = 0)
new_df
```

[79]:

	Bank client ID	First Name	Last Name	Annual Salary
0	1	Mike	Cook	35000
1	2	Sarah	James	85000
2	3	Tim	Fista	55000
3	4	Doran	Hamilton	85000

```
4              5        Pam     Greer          95000
5              6        Tom     Crook          45000
6              7       Bill      Nick          66000
7              8        Jim    Brooks          50000
8              9       Dora      Jade          42000
9             10        Sam     Smith          37000
0             11    Marland  Hamilton          72000
```

14 EXCELLENT JOB!

15 MINI CHALLENGES SOLUTIONS

MINI CHALLENGE #1 SOLUTION: - Write a code that creates the following 2x4 numpy array

```
[[3 7 9 3]
[4 3 2 2]]
```

[ ]:
```
x = np.array([[[3, 7, 9, 3] , [4, 3, 2, 2]]])
x
```

MINI CHALLENGE #2 SOLUTION: - Write a code that takes in a positive integer "x" from the user and creates a 1x10 array with random numbers ranging from 0 to "x"

[ ]:
```
x = int(input("Please enter a positive integer value: "))
x = np.random.randint(1, x, 10)
x
```

[ ]:
```

```

MINI CHALLENGE #3 SOLUTION: - Given the X and Y values below, obtain the distance between them

```
X = [5, 7, 20]
Y = [9, 15, 4]
```

[ ]:
```
X = np.array([5, 7, 20])
Y = np.array([9, 15, 4])
Z = np.sqrt(X**2 + Y**2)
Z
```

MINI CHALLENGE #4 SOLUTION: - In the following matrix, replace the last row with 0

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[ ]:
```
X = np.array([[2, 30, 20, -2, -4],
    [3, 4,  40, -3, -2],
    [-3, 4, -6, 90, 10],
    [25, 45, 34, 22, 12],
    [13, 24, 22, 32, 37]])
```

[ ]:
```
X[4] = 0
X
```

MINI CHALLENGE #5 SOLUTION: - In the following matrix, replace negative elements by 0 and replace odd elements with -2

```
X = [2 30 20 -2 -4]
    [3 4  40 -3 -2]
    [-3 4 -6 90 10]
    [25 45 34 22 12]
    [13 24 22 32 37]
```

[ ]:
```
X = np.array([[2, 30, 20, -2, -4],
    [3, 4,  40, -3, -2],
    [-3, 4, -6, 90, 10],
    [25, 45, 34, 22, 12],
    [13, 24, 22, 32, 37]])

X[X<0] = 0
X[X%2==1] = -2
X
```

MINI CHALLENGE #6 SOLUTION: - A porfolio contains a collection of securities such as stocks, bonds and ETFs. Define a dataframe named 'portfolio_df' that holds 3 different stock ticker symbols, number of shares, and price per share (feel free to choose any stocks) - Calculate the total value of the porfolio including all stocks

[ ]:
```
portfolio_df = pd.DataFrame({'stock ticker symbols':['AAPL', 'AMZN', 'T'],
                             'price per share [$]':[3500, 200, 40],
                             'Number of stocks':[3, 4, 9]})
portfolio_df
```

[ ]:
```
stocks_dollar_value = portfolio_df['price per share [$]'] *␣
 ↪portfolio_df['Number of stocks']
print(stocks_dollar_value)
print('Total portfolio value = {}'.format(stocks_dollar_value.sum()))
```

MINI CHALLENGE #7 SOLUTION: - Write a code that uses Pandas to read tabular US retirement data - You can use data from here: https://www.ssa.gov/oact/progdata/nra.html

[ ]:
```
# Read tabular data using read_html
retirement_age_df = pd.read_html('https://www.ssa.gov/oact/progdata/nra.html')
```

```
retirement_age_df
```

MINI CHALLENGE #8 SOLUTION: - Using "bank_client_df" DataFrame, leverage pandas operations to only select high networth individuals with minimum $5000 - What is the combined networth for all customers with 5000+ networth?

[ ]:
```
df_high_networth = bank_client_df[ (bank_client_df['Net worth [$]'] >= 5000) ]
df_high_networth
```

[ ]:
```
df_high_networth['Net worth [$]'].sum()
```

MINI CHALLENGE #9 SOLUTION: - Define a function that triples the stock prices and adds $200 - Apply the function to the DataFrame - Calculate the updated total networth of all clients combined

[ ]:
```
def networth_update(balance):
    return balance * 3 + 200
```

[ ]:
```
# You can apply a function to the DataFrame
results = bank_client_df['Net worth [$]'].apply(networth_update)
results
```

[ ]:
```
results.sum()
```

PROJECT SOLUTION:

[ ]:
```
# Creating a dataframe from a dictionary
# Let's define a dataframe with a list of bank clients with IDs = 1, 2, 3, 4, 5

raw_data = {'Bank Client ID': ['1', '2', '3', '4', '5'],
            'First Name': ['Nancy', 'Alex', 'Shep', 'Max', 'Allen'],
            'Last Name': ['Rob', 'Ali', 'George', 'Mitch', 'Steve']}

Bank_df_1 = pd.DataFrame(raw_data, columns = ['Bank Client ID', 'First Name',␣
 ↪'Last Name'])
Bank_df_1


# Let's define another dataframe for a separate list of clients (IDs = 6, 7, 8,␣
 ↪9, 10)
raw_data = {
        'Bank Client ID': ['6', '7', '8', '9', '10'],
        'First Name': ['Bill', 'Dina', 'Sarah', 'Heather', 'Holly'],
        'Last Name': ['Christian', 'Mo', 'Steve', 'Bob', 'Michelle']}
Bank_df_2 = pd.DataFrame(raw_data, columns = ['Bank Client ID', 'First Name',␣
 ↪'Last Name'])
Bank_df_2
```

```
# Let's assume we obtained additional information (Annual Salary) about our␣
 ↪bank customers
# Note that data obtained is for all clients with IDs 1 to 10
raw_data = {
        'Bank Client ID': ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'],
        'Annual Salary [$/year]': [25000, 35000, 45000, 48000, 49000, 32000,␣
 ↪33000, 34000, 23000, 22000]}
bank_df_salary = pd.DataFrame(raw_data, columns = ['Bank Client ID','Annual␣
 ↪Salary [$/year]'])
bank_df_salary


# Let's concatenate both dataframes #1 and #2
# Note that we now have client IDs from 1 to 10
bank_df_all = pd.concat([Bank_df_1, Bank_df_2])
bank_df_all


# Let's merge all data on 'Bank Client ID'
bank_df_all = pd.merge(bank_df_all, bank_df_salary, on = 'Bank Client ID')
bank_df_all
```

[ ]:
```
new_client = {
        'Bank Client ID': ['11'],
        'First Name': ['Ry'],
        'Last Name': ['Aly'],
        'Annual Salary [$/year]' : [1000]}
new_client_df = pd.DataFrame(new_client, columns = ['Bank Client ID', 'First␣
 ↪Name', 'Last Name', 'Annual Salary [$/year]'])
new_client_df
```

[ ]:
```
new_df = pd.concat([bank_df_all, new_client_df], axis = 0)
new_df
```